BAB III
ANALISA DAN PERANCANGAN SISTEM

3.1 Analisa Masalah

Tujuan utama improvisasi algoritma RSA adalah untuk memperbaiki perfoma algoritma tersebut, hal ini secara mendetail sudah dibahas pada sub bab 2.3.3. Perfoma algoritma RSA dapat dilihat melalui waktu yang diperlukan untuk melakukan pembangkitan kunci, enkripsi dan juga dekripsi. Pengujian perfoma antara algoritma RSA standar dan improvisasi algoritma RSA dilakukan untuk menganalisis perbandingan perfoma antar kedua algoritma tersebut.

Algoritma RSA memiliki banyak celah keamanan yang memungkinkan terjadinya berbagai jenis metode penyerangan. Jenis-jenis metode serangan ini sudah dipaparkan pada sub bab 2.3.2. Tujuan utama penggunaan algoritma RSA adalah untuk keamanan pengguna. Algoritma RSA mengalami banyak improvisasi untuk memperbaiki perfomanya misalnya seperti mengurangi biaya dekripsi dengan menerapkan metode CRT (*Chinese Remainder Theorem*). Celah keamanan algoritma RSA tentu juga harus diperhatikan selain memperbaiki perfoma algoritma tersebut. Hasil improvisasi algoritma RSA diharapkan mampu menutupi celah keamanan pada algoritma RSA. Untuk memastikan hal ini, pengujian keamanan dilakukan. Pengujian keamanan antara algoritma RSA standar dan improvisasi algoritma RSA dipergunakan untuk menganalisis perbandingan keamanan antara kedua algoritma tersebut. Metode uji keamanan yang digunakan adalah *known plain text attack* dan *wiener attack*. Hasil pengujian perfoma dan pengujian keamanan diharapkan dapat menjadi bahan evaluasi untuk mengukur kualitas hasil improvisasi algoritma RSA baik dari segi perfoma maupun keamanan.

3.2 Rancangan Algoritma RSA Standar

Rancangan algoritma RSA standar ini sesuai dengan algoritma RSA asli pada sub bab 2.3.1. Rancangan ini nantinya diimplementasikan pada program komputer yang ditulis dengan bahasa pemrograman java. Algoritma ini terbagi menjadi tiga mekanisme yang berbeda yang dilakukan secara berurutan.

3.2.1 Pembangkitan kunci algoritma RSA standar

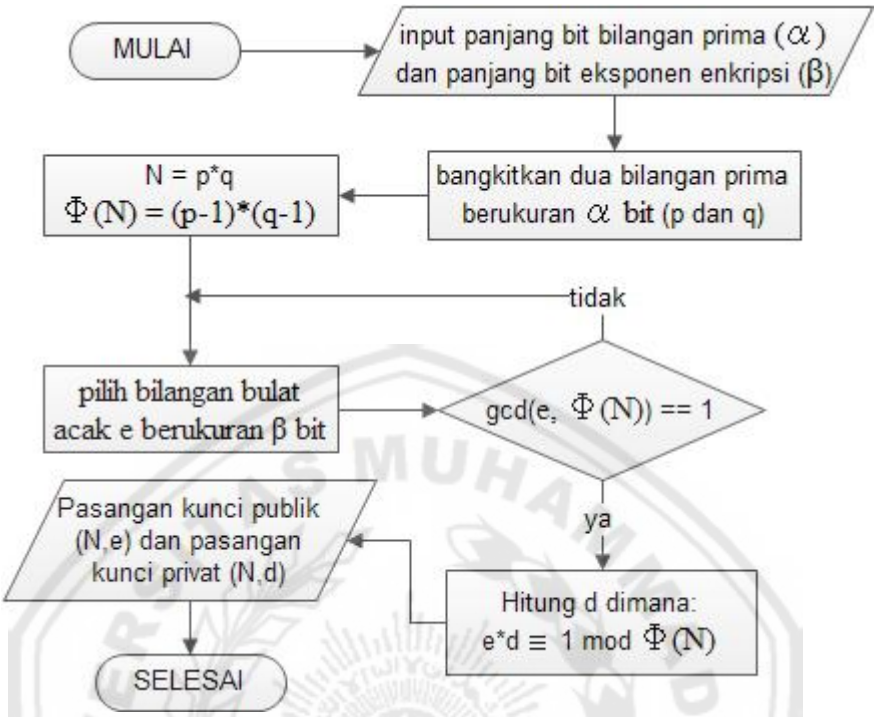
Pembangkitan kunci pada algoritma RSA standar diawali dengan membangkitkan dua bilangan prima (p dan q). Pembangkitan bilangan prima ini melibatkan algoritma pengujian bilangan prima. Algoritma yang digunakan untuk pengujian bilangan prima pada penelitian ini adalah algoritma Miller-Rabin. Jika dua bilangan prima (p dan q) sudah terpilih dilanjutkan menghitung nilai N dengan rumus 2.1 dan menghitung Φ(N) dengan rumus 2.2. Langkah selanjutnya adalah memilih bilangan bulat e secara acak dengan ketentuan faktor persekutuan terbesar (FPB) atau *great common divisor (gcd)* antara e dan Φ(N) sama dengan satu (lihat rumus 2.3). Jika nilai e sudah terpilih dilanjutkan dengan perhitungan nilai d. Hubungan nilai e dan d dituliskan seperti pada rumus 2.4. Pasangan nilai e dan N menjadi kunci publik yang selanjutnya didistribusikan untuk melakukan enkripsi sedangkan pasangan nilai d dan N menjadi kunci privat yang dirahasiakan untuk melakukan dekripsi. Pseudo code proses pembangkitan kunci algoritma RSA standar adalah sebagai berikut:

Input: panjang bit bilangan prima (α), panjang bit eksponen enkripsi (β)
Output: K_{publik}, K_{privat}
1. Bangkitkan dua bilangan prima berukuran α bit (p dan q)
2. $N = p \times q$
3. $\Phi(N) = (p - 1) \times (q - 1)$
4. Pilih bilangan bulat e berukuran β bit dengan gcd(e, Φ(N)) = 1, $1 < e < \Phi(N)$
5. $d = e^{-1} \bmod \Phi(N)$
6. $K_{publik} = (e, N)$, $K_{privat} = (d, N)$

Gambar 3.1: pseudocode pembangkitan kunci algoritma RSA standar

Proses pembangkitan kunci algoritma RSA standar membutuhkan input panjang bit bilangan prima dan panjang bit eksponen enkripsi. Input tersebut ditentukan oleh user sesuai dengan pilihan yang sudah disediakan. Panjang bit bilangan prima yang digunakan pada penelitian ini adalah 128 bit dan 256 bit, sedangkan panjang bit eksponen enkripsi yang digunakan pada penelitian ini antara lain berukuran sama dengan bilangan prima (default), panjang bit nilai Φ(N) − 4, panjang bit nilai Φ(N) − 3, dan panjang bit nilai Φ(N) − 2. User dapat

memilih salah satu panjang bit bilangan prima dan panjang bit eksponen tersebut. Alur proses pembangkitan kunci algoritma RSA standar adalah sebagai berikut:



Gambar 3.2: alur pembangkitan kunci algoritma RSA standar

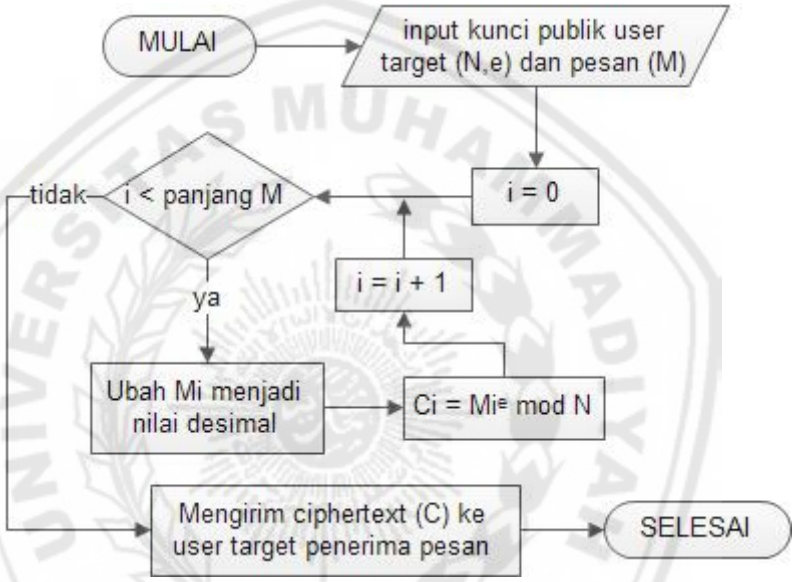
3.2.2 Enkripsi RSA standar

Enkripsi dipergunakan untuk menyandikan pesan sebelum pesan tersebut dikirim melalui saluran jaringan. Sebelum pesan dienkripsi, masing-masing karakter pada pesan diubah menjadi nilai ASCII desimal. Tiap-tiap nilai ASCII ini kemudian diubah menjadi nilai cipher text (C) dengan rumus 2.5 dan digabungkan menjadi satu membentuk pesan terenkripsi. Pseudo code proses enkripsi algoritma RSA standar adalah sebagai berikut:

Input: K_{publik} = (e, N), plain text (M)
Output: cipher text (C)
1. for i ← 0 to PanjangPlaintext − 1 do
2. Konversi M_i ke nilai desimal
3. $C_i \leftarrow (M_i)^e \bmod N$
4. end for
5. Return C

Gambar 3.3: pseudocode enkripsi algoritma RSA standar

Proses enkripsi memerlukan input kunci publik dari user target penerima pesan dan pesan asli atau plain text (M). User menginput sebuah pesan singkat dan memilih user target penerima pesan. Setiap user memiliki kunci publik yang disebarluaskan melalui saluran jaringan. Nama user beserta kunci publiknya disimpan dalam daftar user online. Daftar user online dimiliki oleh setiap user. Ketika user memilih user target penerima pesan maka sistem akan otomatis menginput kunci publik user tersebut untuk proses enkripsi. Alur proses enkripsi algoritma RSA standar adalah sebagai berikut:



Gambar 3.4: alur enkripsi algoritma RSA standar

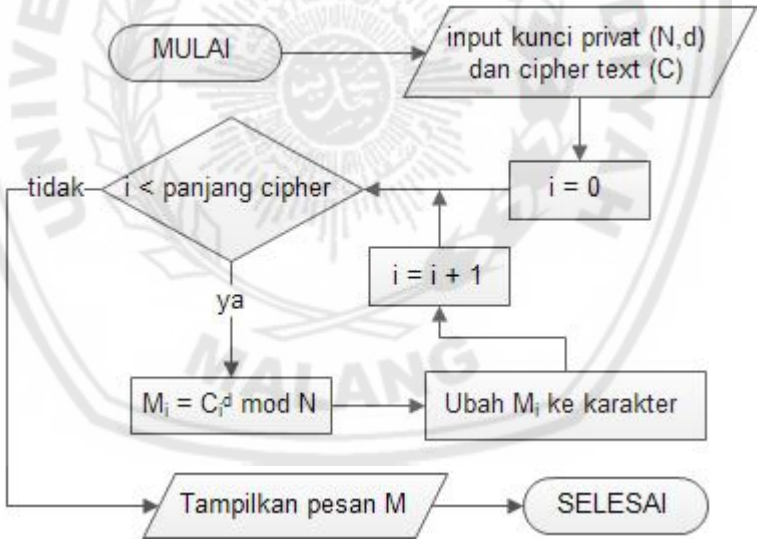
3.2.3 Dekripsi Algoritma RSA

Dekripsi digunakan untuk mengubah pesan terenkripsi (*cipher text* atau C) menjadi sebuah pesan asli (*plain text* atau M). Nilai M dihitung dengan rumus 2.6 untuk tiap nilai C yang ada pada pesan terenkripsi. Nilai M hasil dekripsi berupa nilai desimal yang kemudian diubah menjadi karakter. Jika pada enkripsi nilai desimal karakter diperoleh dari nilai ASCII desimal maka pada dekripsi pengubahan nilai M ke karakter juga dilakukan berdasarkan nilai ASCII. Tiap karakter hasil pengubahan tersebut disatukan kembali sehingga membentuk pesan asli. Pseudo code dekripsi algoritma RSA standar adalah sebagai berikut:

Input: K_{privat} = (d, N), cipher text (C)
Output: plain text (M)
1. for i ← 0 to PanjangCiphertext − 1 do
2. $M_i \leftarrow (C_i)^d \bmod N$
3. Konversi nilai M_i ke karakter
4. end for
5. Return M

Gambar 3.5: pseudocode dekripsi algoritma RSA standar

Proses dekripsi algoritma RSA standar memerlukan input kunci privat (N,d) dan pesan terenkripsi (C). Setiap user memiliki satu kunci privat yang diperoleh setelah melakukan pembangkitan kunci. Kunci privat tersebut tersimpan secara rahasia dan tidak didistribusikan kepada siapapun. Proses dekripsi dilakukan setiap kali user menerima pesan baru dari user lain. Sistem menginputkan pesan terenkripsi dan kunci privat secara otomatis untuk melakukan dekripsi. Alur proses dekripsi algoritma RSA standar adalah sebagai berikut:



Gambar 3.6: alur dekripsi algoritma RSA standar

3.3 Rancangan Improvisasi Algoritma RSA

Rancangan improvisasi algoritma RSA ini mengikuti usulan improvisasi yang digagas oleh Nikita Somani dan Dharmendra Mangal. Usulan improvisasi tersebut sudah dibahas pada sub bab 2.3.3. Perbedaannya terletak pada variasi jumlah bilangan prima pada proses pembangkitan kunci dan metode dekripsi.

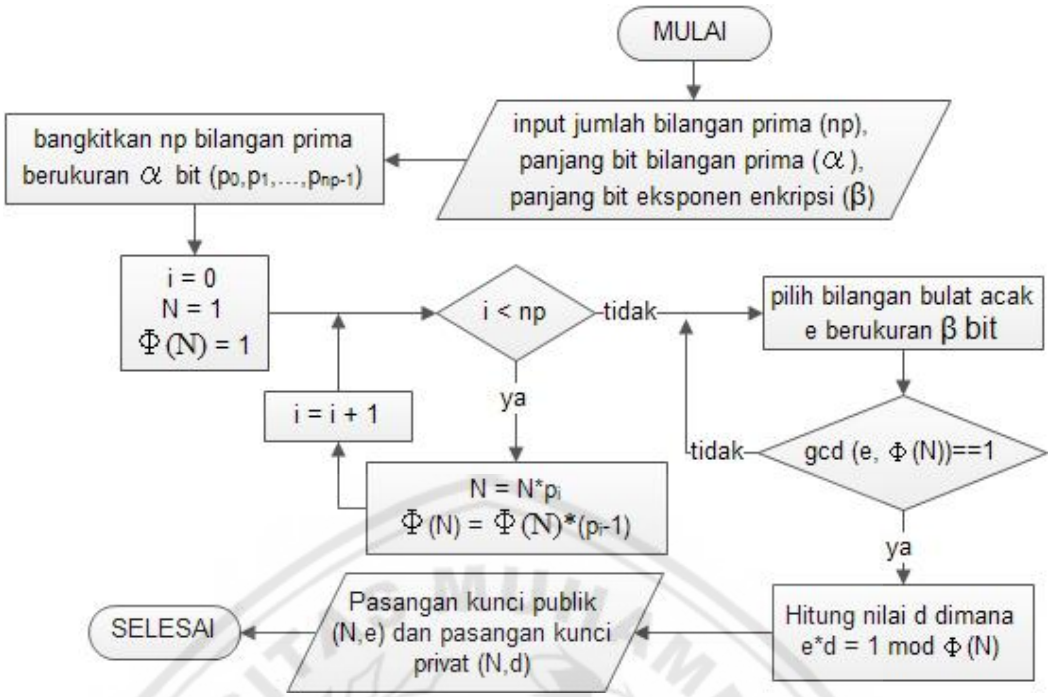
3.3.1 Pembangkitan kunci improvisasi algoritma RSA

Proses pembangkitan kunci improvisasi algoritma RSA diawali dengan membangkitkan sejumlah bilangan prima yang berbeda. Langkah berikutnya adalah menhitung nilai N dengan rumus 2.11 dan menghitung nilai Φ(N) dengan rumus 2.12. Improvisasi algoritma RSA ini menggunakan 2 hingga 6 bilangan prima sehingga perhitungan nilai N dan Φ(N) disesuaikan dengan jumlah penggunaan bilangan prima. Misalkan jumlah bilangan prima yang dibangkitakan adalah empat maka untuk mencari nilai N keempat bilangan prima tersebut dikalikan sedangkan untuk mencari nilai Φ(N) tiap bilangan prima dikurangi satu kemudian dikalikan keempatnya. Langkah selanjutnya adalah memilih nilai e dengan rumus 2.13 dan menghitung nilai d dengan rumus 2.14. Penentuan kunci publik dan kunci privat juga sama dengan algoritma RSA standar. Pseudo code pembangkitan kunci improvisasi algoritma RSA adalah sebagai berikut:

Input: jumlah bilangan prima (np), panjang bit bilangan prima (α), panjang bit eksponen enkripsi (β)
Output: K_{publik}, K_{privat}
1. Bangkitkan np bilangan prima (p_0, p_1, …, p_{np-1})
2. N = 1
3. Φ(N) = 1
4. for i ← 0 to np − 1 do
5. $N = N \times p_i$
6. $\Phi(N) = \Phi(N) \times (p_i - 1)$
7. end for
8. Pilih bilangan bulat e dengan gcd(e, Φ(N)) = 1, 1 < e < Φ(N)
9. $d = e^{-1} \bmod \Phi(N)$
10. $K_{publik} = (e, N)$, $K_{privat} = (d, N)$

Gambar 3.7: pseudocode pembangkitan kunci improvisasi algoritma RSA

Proses pembangkitan kunci improvisasi algoritma RSA memerlukan input jumlah bilangan prima, panjang bit bilangan prima, dan panjang bit eksponen enkripsi. Input tersebut ditentukan oleh user dengan pilihan yang sudah disediakan. Jumlah bilangan prima yang digunakan pada penelitian ini adalah 2 hingga 6 bilangan prima. Panjang bit bilangan prima dan eksponen enkripsi sama dengan algoritma RSA standar. Alur proses pembangkitan kunci improvisasi algoritma RSA adalah sebagai berikut:



Gambar 3.8: alur pembangkitan kunci improvisasi algoritma RSA

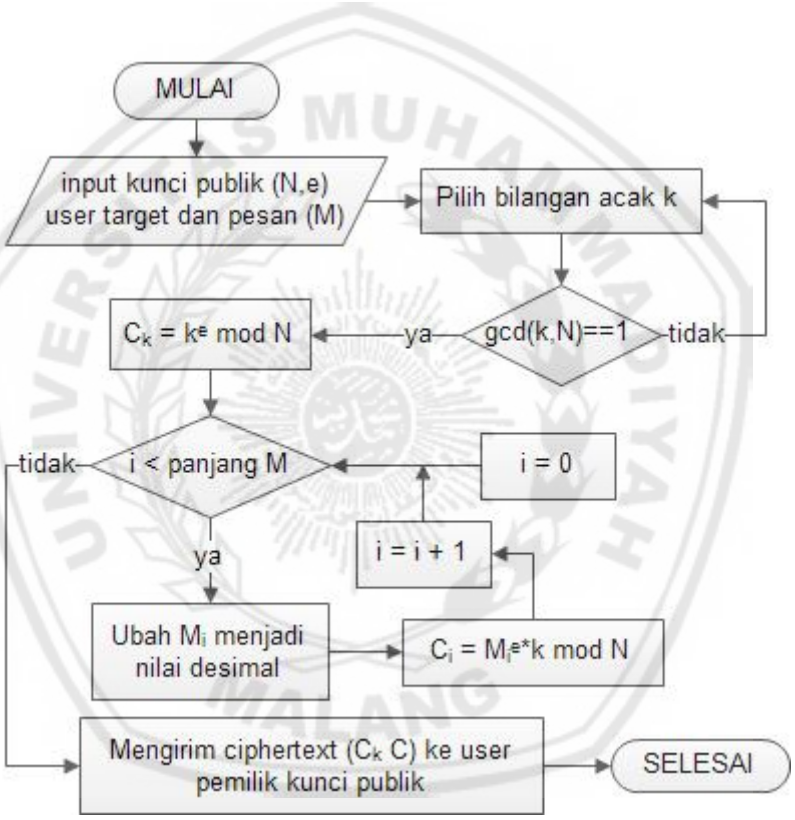
3.3.2 Enkripsi improvisasi algoritma RSA

Proses enkripsi pada improvisasi algoritma RSA diawali dengan memilih bilangan k secara acak dengan rumus 2.15. Nilai k ini kemudian dienkripsi menggunakan kunci publik (N,e) user target penerima pesan menjadi nilai Ck dengan rumus 2.16. Proses berikutnya adalah mengubah nilai ASCII desimal tiap karakter menjadi nilai *cipher text* (C) dengan rumus 2.17. Nilai Ck dan C tiap karakter digabung menjadi satu membentuk pesan terenkripsi. Pseudo code enkripsi improvisasi algoritma RSA adalah sebagai berikut:

Input: K_{publik} = (e, N), plain text (M)
Output: cipher text (C), Ck
1. Pilih bilangan acak k dengan gcd(k, N) = 1, 1 < k < N
2. $C_k = k^e \bmod N$
3. for i ← 0 to PanjangPlaintext − 1 do
4. Konversi Mi ke nilai desimal
5. $C_i \leftarrow (M_i)^e * k \bmod N$
6. end for
7. Return C, Ck

Gambar 3.9: pseudo code enkripsi improvisasi RSA standar

Proses enkripsi improvisasi algoritma RSA memerlukan input yang sama dengan input proses enkripsi algoritma RSA standar. Input tersebut akan dimasukkan ke sistem ketika user sudah menginput pesan singkat dan memilih user target penerima pesan dari daftar user online. Rumus yang digunakan untuk menghitung cipher text tiap karakter pada improvisasi algoritma RSA berbeda dengan algoritma RSA standar. Output dari proses enkripsi improvisasi algoritma RSA adalah cipher dari nilai k (C_k) dan pesan terenkripsi (C). Output ini dikirimkan sekaligus ke user target penerima pesan. Alur dari proses enkripsi improvisasi algoritma RSA adalah sebagai berikut:



Gambar 3.10: alur proses enkripsi improvisasi algoritma RSA

3.3.3 Dekripsi improvisasi algoritma RSA

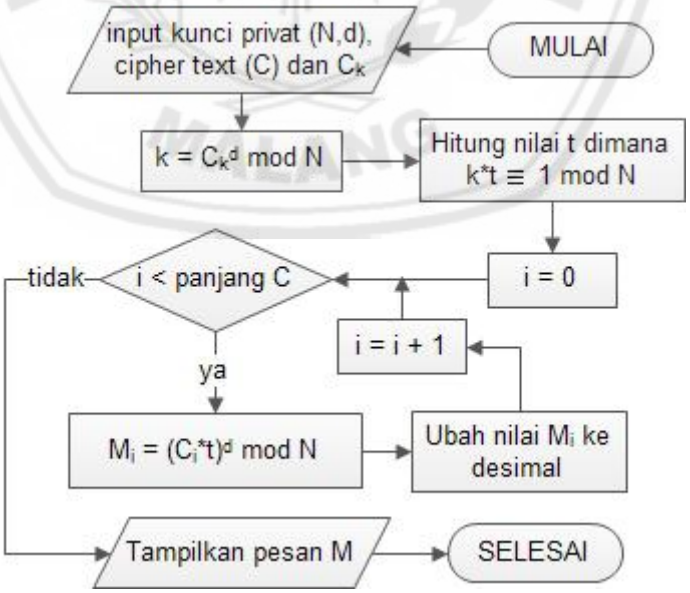
Metode dekripsi yang digunakan dalam model improvisasi algoritma RSA ini berbeda dengan metode dekripsi pada usulan improvisasi algoritma RSA yang digagas oleh Nikita Somani dan Dharmendra Mangal pada sub bab 2.3.3. Langkah pertama adalah mendekrip nilai k dari cipher text k (C_k) dengan menggunakan rumus 2.18. Kemudian dilanjutkan dengan menghitung nilai t. Nilai

t dihitung dengan menggunakan rumus 2.19. Jika nilai t sudah dihitung baru dilanjutkan dengan menghitung nilai *plain text* (M) dari masing-masing *cipher text* (C) dengan menggunakan rumus 2.20. Pseudo code dekripsi improvisasi algoritma RSA adalah sebagai berikut:

Input: K_{privat} = (d, N), cipher text (C), C_k
Output: plain text (M)
1. k = C_k^d mod N
2. t = k^{-1} mod N
3. for i ← 0 to PanjangCiphertext − 1 do
4. $M_i \leftarrow (C_i * t)^d$ mod N
5. Konversi nilai M_i ke karakter
6. end for
7. Return M

Gambar 3.11: pseudo code dekripsi improvisasi algoritma RSA

Proses dekripsi improvisasi algoritma RSA memerlukan input kunci privat (N,d), pesan terenkripsi (C) dan cipher k (Ck). Input ini akan secara otomatis terisi pada sistem ketika user mendapat pesan baru atau pesan masuk dari user lain. Nilai k hasil dekripsi nilai Ck dipergunakan untuk menghitung nilai M pada tiap nilai C pada pesan terenkripsi. Alur dari proses dekripsi adalah sebagai berikut:



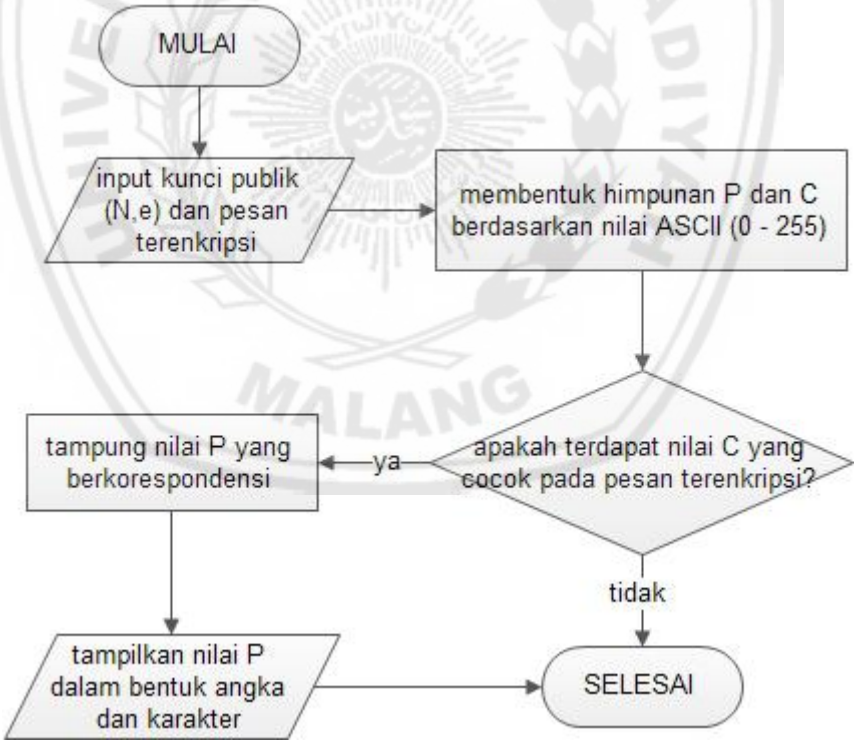
Gambar 3.12: alur dekripsi improvisasi algoritma RSA

3.4 Rancangan Uji Keamanan

Algoritma RSA memiliki banyak kerentanan yang dapat dipergunakan untuk melumpuhkan keamanannya. Informasi yang didapatkan dari hasil metode penyerangan ketika berhasil diantaranya adalah menemukan pesan asli (M) tanpa menggunakan nilai eksponen dekripsi (d) dan menemukan nilai eksponen dekripsi (d) pada kunci privat (N,d) untuk mendekripsi *cipher text* (C). Rancangan uji keamanan ini dipergunakan untuk menganalisis perbandingan keamanan antara algoritma RSA standar dan improvisasi algoritma RSA.

3.4.1 Known plain text attack

Known plain text attack termasuk metode penyerangan yang digunakan menemukan *plain text* (M) dari *cipher text* (C) tanpa menggunakan nilai eksponen dekripsi (d). Known plain text attack ini sudah dibahas pada sub bab 2.3.2.2. Alur proses *known plain text attack* adalah sebagai berikut:



Gambar 3.13: alur proses known plain text attack

Proses known plain text attack diawali dengan menginputkan kunci publik dan pesan terenkripsi atau cipher text. Proses selanjutnya adalah membentuk

himpunan plain text dan cipher text. Himpunan plain text diisi dengan bilangan ASCII desimal mulai dari 0 hingga 255. Tiap nilai pada himpunan plain text dihitung nilai cipher text (C). Jika yang diuji adalah algoritma RSA standar maka perhitungannya menggunakan rumus 2.5, sedangkan jika yang diuji adalah improvisasi algoritma RSA maka perhitungannya menggunakan rumus 2.16 untuk mencari *cipher text* k dan rumus 2.17 untuk mencari *cipher text*. Setiap plain text yang ditemukan ditampung terlebih dahulu sebelum ditampilkan. Sistem akan menampilkan plain text dalam bentuk angka ASCII desimal dan dalam bentuk karakter sesuai bilangan ASCII yang mempresentasikannya jika metode *known plain text attack* berhasil meretasnya.

3.4.2 Wiener Attack

Wiener attack dapat digunakan untuk menemukan eksponen dekripsi (d) dari sepasang nilai kunci publik (N,e) ketika nilai $d < N^{1/4}$, hal ini sudah dibahas pada sub bab 2.3.2.1. Wiener attack memiliki beberapa urutan langkah. Langkah awal yang harus dikerjakan adalah membuat fraksaksi berkelanjutan (q0, q1, …, qn) dari sepasang nilai e dan N dengan rumus 2.7. Pseudo code untuk membentuk fraksaksi berkelanjutan adalah sebagai berikut:

Input: K_{publik} (N,e)
Output: fraksaksi berkelanjutan (q_0, q_1, …, q_n)
1. $q_0 = e/N$
2. r = e mod N
3. e = N
4. N = r
5. while r != 0 do
6. $q_{i+1} = e/N$
7. r = e mod N
8. e = N
9. N = r
10. end while

Gambar 3.14 pseudo code fraksaksi berkelanjutan

Proses pembentukan fraksaksi berkelanjutan memerlukan input kunci publik yang terdiri dari sepasang nilai e dan N. Proses pembentukannya dilakukan dengan membagi nilai e dengan N. Jumlah elemen fraksaksi berkelanjutan tergantung

pada sebarapa banyak proses pembagian yang dilakukan hingga mendapat sisa pembagian sama dengan nol. Fraksaksi berkelanjutan ini digunakan untuk membentuk fraksaksi k/d dengan rumus 2.8. Pseudo code pembentukan fraksaksi k/d adalah sebagai berikut:

Input: fraksaksi berkelanjutan (q_0, q_1, …, q_n)
Output: fraksaksi k/d
1. $k_0 = q_0$
2. $d_0 = 1$
3. $k_1 = (q_0 * q_1)+1$
4. $d_1 = q_1$
5. for i ← 2 to Jumlah elemen q do
6. $k_i = (q_i * k_{i-1})+k_{i-2}$
7. $d_i = (q_i * d_{i-1})+d_{i-2}$
8. end for

Gambar 3.15 pseudo code fraksaksi k/d

Proses pembentukan fraksaksi k/d memerlukan input fraksaksi berkelanjutan (q0, q1, …, qn) yang dibentuk pada proses sebelumnya. Setiap nilai k dan d pada fraksaksi k/d ini nantinya digunakan untuk menemukan faktor dari nilai N pada kunci publik. Jika berhasil maka nilai d dapat ditemukan. Pseudo code wiener attack secara keseluruhan adalah sebagai berikut:

Input: K_{publik} (N,e)
Output: eksponen dekripsi (d)
1. membuat fraksaksi berkelanjutan (q_0, q_1, …, q_n)
2. membuat fraksaksi k/d
3. for i ← 1 to jumlah fraksaksi k/d do
4. $\Phi(N) = ((e * d_i) - 1) / k_i$
5. if ($\Phi(N)$!= 0) do
6. $b = N - \Phi(N) + 1$
7. $D = b^2 - 4*N$
8. $x1 = (b + \sqrt{D}) / 2$
9. $x2 = (b - \sqrt{D}) / 2$
10. if (x1*x2 == N)
11. $d = e^{-1} \bmod \Phi(N)$
12. return d
13. end if
14. end if
15. end for

Gambar 3.16 pseudo code wiener attack

3.5 Perbandingan algoritma RSA standar dan improvisasi algoritma RSA

Contoh cara kerja dari algoritma RSA dan improvisasi algoritma RSA perlu dibahas terlebih dahulu sebelum mengarah ke pembahasan perbandingan dari kedua algoritma tersebut. Misalkan Alice akan mengirimkan pesan kepada Bob melalui saluran jaringan publik. Hal pertama yang dilakukan oleh Bob adalah membangkitkan sepasang kunci terlebih dahulu melalui mekanisme pembangkitan kunci. Contoh pembangkitan kunci pada algoritma RSA standar adalah sebagai berikut:

1. Bilangan prima yang terpilih p = 227 dan q = 113
2. Sesuai rumus 2.1 nilai N = 227*113 = 25651
3. Sesuai rumus 2.2 nilai Φ(N) = (227−1)*(113−1) = 226*112 = 25312.
4. Sesuai rumus 2.3 eksponen enkripsi yang terpilih, e = 16063 dengan gcd(16063, 25312) = 1.
5. Sesuai rumus 2.4 eksponen dekripsi d = 13939, 16063*13939 ≡ 1 mod 25312.
6. Kunci publik (25651, 16063) dan kunci privat (25651, 13939).

Sedangkan untuk contoh pembangkitan kunci pada improvisasi algoritma RSA dengan menggunakan 4 bilangan prima adalah sebagai berikut:

1. Bilangan yang terpilih $p_1 = 149$, $p_2 = 223$, $p_3 = 151$, dan $p_4 = 229$.
2. Sesuai rumus 2.11 N = 149*223*151*229 = 1148956433
3. Sesuai rumus 2.12 Φ(N) = (149 − 1)*(223 − 1)*(151 − 1)*(229 − 1) = 1123675200.
4. Sesuai rumus 2.13 eksponen enkripsi yang terpilih e = 544104769 dengan gcd(544104769, 1123675200) = 1.
5. Sesuai rumus 2.14 eksponen dekripsi d = 90400129, 544104769*90400129 ≡ 1 mod 1123675200. Kunci publik (1148956433, 544104769) dan kunci privat (1148956433, 90400129).

Setelah Bob melakukan pembangkitkan kunci, kunci publiknya dikrimkan ke Alice untuk mengenkripsi pesan. Contoh enkripsi pesan menggunakan algoritma RSA standar adalah sebagai berikut:

1. Memperoleh kunci publik dari Bob (25651, 16063).
2. Alice akan mengirimkan pesan "halo". Tiap karakter dalam pesan tersebut diubah menjadi nilai desimal misalnya nilai ASCII desimal dari tiap karakter tersebut. Nilai ASCII desimal tiap karakter dalam pesan tersebut adalah 104, 97, 108 dan 111.
3. Nilai desimal dalam tiap karakter (M) kemudian dienkripsi dengan menggunakan kunci publik Bob dengan perhitungan sebagai berikut (gunakan rumus 2.5):
   $C_1 = 104^{16063} \bmod 25651 = 23689$ $C_2 = 97^{16063} \bmod 25651 = 1646$
   $C_3 = 108^{16063} \bmod 25651 = 8725$ $C_4 = 111^{16063} \bmod 25651 = 3874$
4. Mengirimkan hasil enkripsi pesan (23689 1646 8725 3874) kepada Bob.

Jika enkripsinya menggunakan improvisasi algoritma RSA contoh enkripsi pesannya adalah sebagai berikut:

1. Memperoleh kunci publik dari Bob (1148956433, 544104769).
2. Alice akan mengirimkan pesan "halo". Tiap karakter dalam pesan tersebut diubah menjadi nilai desimal misalnya nilai ASCII desimal dari tiap karakter tersebut. Nilai ASCII desimal tiap karakter dalam pesan tersebut adalah 104, 97, 108 dan 111.
3. Sesuai rumus 2.15 bilangan acak k yang terpilih adalah k = 48139141 dengan gcd(48139141, 1148956433) = 1 (gunakan rumus 2.26).
4. Mengenkripsi nilai k menggunakan rumus 2.16 sehingga Ck = $48139141^{544104769} \bmod 1148956433 = 22232829$ (gunakan rumus 2.27).
5. Nilai desimal dalam tiap karakter (M) kemudian dienkripsi dengan menggunakan kunci publik Bob dan nilai k dengan perhitungan sebagai berikut (gunakan rumus 2.17):
   $C_1 = 104^{544104769} * 48139141 \bmod 1148956433 = 1145427828$
   $C_2 = 97^{544104769} * 48139141 \bmod 1148956433 = 748431701$

$C_3 = 108^{544104769} * 48139141 \bmod 1148956433 = 616598036$

$C_4 = 111^{544104769} * 48139141 \bmod 1148956433 = 243001563$

6. Mengirimkan hasil enkripsi nilai k beserta hasil enkripsi pesan (22232829 1145427828 748431701 616598036 243001563) kepada Bob.

Setelah Bob berhasil menerima pesan dari Alice, untuk bisa memahami isi pesan tersebut Bob harus melakukan dekripsi terlebih dahulu dengan menggunakan kunci privatnya. Contoh proses dekripsi pesan menggunakan algoritma RSA standar adalah sebagai berikut:

1. Menghitung nilai plain text (M) dari masing-masing cipher text dengan perhitungan sebagai berikut (gunakan rumus 2.6):

   $M_1 = 23689^{13919} \bmod 25651 = 104$ $M_2 = 1646^{13919} \bmod 25651 = 97$

   $M_3 = 8725^{13919} \bmod 25651 = 108$ $M_4 = 3874^{13919} \bmod 25651 = 111$
2. Nilai M dari hasil perhitungan diatas kemudian dikembalikan lagi menjadi bentuk karakter. Jika pada proses enkripsi nilai desimal tiap karakter diperoleh berdasarkan nilai ASCII desimal maka nilai M hasil perhitungan dekripsi diubah menjadi karakter sesuai dengan nilai ASCII yang mempresentasikannya. Berdasarkan nilai ASCII dari tiap nilai M jika digabungkan pesan tersebut adalah “halo”.

Sedangkan contoh proses dekripsi pesan jika menggunakan improvisasi algoritma RSA adalah sebagai berikut:

1. Menghitung nilai k dari nilai cipher text k (Ck) dengan rumus 2.18, $k = 22232829^{90400129} \bmod 1148956433 = 48139141$.
2. Menghitung nilai t dengan rumus 2.19, t = 743843849, $48139141 * 743843849 \equiv 1 \bmod 1148956433$.
3. Menghitung nilai plain text (M) dari masing-masing cipher text dengan perhitungan sebagai berikut (gunakan rumus 2.20):

   $M_1 = (1145427828 * 743843849)^{90400129} \bmod 1148956433 = 104$

   $M_2 = (748431701 * 743843849)^{90400129} \bmod 1148956433 = 97$

$M_3 = (616598036*743843849)^{90400129} \bmod 1148956433 = 108$

$M_4 = (243001563*743843849)^{90400129} \bmod 1148956433 = 111$

4. Nilai M dari hasil perhitungan diatas kemudian dikembalikan lagi menjadi bentuk karakter. Jika pada proses enkripsi nilai desimal tiap karakter diperoleh berdasarkan nilai ASCII desimal maka nilai M hasil perhitungan dekripsi diubah menjadi karakter sesuai dengan nilai ASCII yang mempresentasikannya. Berdasarkan nilai ASCII dari tiap nilai M jika digabungkan pesan tersebut adalah "halo".

Ulasan diatas dapat dijadikan sebagai bahan wawasan untuk mencari perbandingan pada kedua algoritma tersebut sehingga dari contoh cara kerja algoritma diatas dapat diambil beberapa perbedaan antara algoritma RSA standar dan improvisasi algoritma RSA sebagai berikut:

Tabel 3.1: perbedaan algoritma RSA standar dan improvisasi algoritma RSA

Proses	Pembeda	Algoritma RSA standar	Improvisasi algoritma RSA
Pembangkitan kunci	Jumlah bilangan prima yang dibangkitkan	Membangkitkan dua bilangan prima (p dan q)	Membangkitkan 2 hingga 6 bilangan prima (p_1, p_2, …, p_n)
	Perhitungan nilai N dan Φ(N)	$N = p*q$ dan $\Phi(N) = (p-1)*(q-1)$	$N = p_1*p_2*\ldots*p_n$ dan $\Phi(N) = (p_1-1)*(p_2-1)*\ldots*(p_n-1)$
Enkripsi	Penggunaan bilangan acak k	Tidak menggunakan bilangan acak k	Menggunakan bilangan acak k dimana gcd(k,N) = 1
	Perhitungan nilai cipher text (C)	$C = M^e \bmod N$	$C = M^e*k \bmod N$
	Cipher text yang dikirimkan	Hanya mengirimkan nilai C dari tiap karakter dalam	Mengirimkan nilai cipher dari k ($Ck = k^e \bmod N$) dan nilai C tiap karakter dalam

		pesan	pesan.
Dekripsi	Urutan langkah sebelum menghitung nilai plain text (M)	Langsung menghitung nilai M	Menemukan nilai k dengan $k = Ck^d \bmod N$ kemudian mencari nilai t dimana $k*t \equiv 1 \bmod N$ baru setelah itu menghitung nilai M
	Perhitungan nilai plain text (M)	$M = C^d \bmod N$	$M = (C*t)^d \bmod N$